AF ZZW $

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellants:	Michael Freed; Elango Ganesan	Confirmation No.	4136
Serial No.:	09/900,494		
Filed:	July 6, 2001	Customer No.:	28863
Examiner:	Aravind K. Moorthy		
Group Art Unit:	2131		
Docket No.:	1014-064US01/JNP-0261		
Title:	LOAD BALANCING SECURE SOCKETS LAYER ACCELERATOR		

CERTIFICATE UNDER 37 CFR 1.8: I hereby certify that this correspondence is being deposited with the United States Post Service, as First Class Mail, in an envelope addressed to: Commissioner for Patents, Alexandria, VA 22313-1450 on December 20, 2006.

By: [illegible]
Name: Caryl Harriman

Mail Stop Appeal Brief – Patents
Board of Patent Appeals and Interferences
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313

Sir:

We are transmitting herewith the attached correspondence relating to this application:

- ☒ Transmittal sheet containing Certificate of Mailing
- ☒ Brief on Appeal in triplicate (32 pgs.)
- ☒ Check in the amount of $500.00 for required fee for Appeal Brief
- ☒ Return postcard

Please apply any charges not covered, or any credits, to Deposit Account No. 50-1778.

Date:
December 20, 2006

By: [illegible]
Name: Jennifer M.K. Rogers
Reg. No.: 58,695

SHUMAKER & SIEFFERT, P.A.
8425 Seasons Parkway, Suite 105
St. Paul, Minnesota 55125
Telephone: 651.735.1100
Facsimile: 651.735.1102

PATENT

IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

Appellants:	Michael Freed; Elango Ganesan	Confirmation No.	4136
Serial No.:	09/900,494		
Filed:	July 6, 2001	Customer No.:	28863
Examiner:	Aravind K. Moorthy		
Group Art Unit:	2131		
Docket No.:	1014-064US01/JNP-0261		
Title:	LOAD BALANCING SECURE SOCKETS LAYER ACCELERATOR		

APPEAL BRIEF

Mail Stop Appeal Brief – Patents
Board of Patent Appeals and Interferences
Commissioner for Patents
P.O. Box 1450
Alexandria, VA 22313

Dear Sir:

This is an Appeal Brief from the Final Office Action mailed April 20, 2006. The Notice of Appeal was filed on August 23, 2006. The present application has been rejected more than twice. Appellants submit this Appeal Brief in triplicate. Enclosed is a check in the amount of $500.00 for the fee for filing a brief in support of an appeal.

Appellants request the opportunity for a personal appearance before the Board of Patent Appeals and Interferences to argue the issues of this appeal. The fee for the personal appearance will be timely paid upon receipt of the Examiner's Answer. Please charge any additional fees that may be required or credit any overpayment to Deposit Account No. 50-1778.

```
12/29/2006 WASFAW1  00000042 09900494
01 FC:1402                   500.00 OP
```

TABLE OF CONTENTS

REAL PARTY IN INTEREST

The real party in interest is Juniper Networks, Inc. of Sunnyvale, California.

RELATED APPEALS AND INTERFERENCES

There are no related appeals and interferences.

STATUS OF CLAIMS

Claims 1–28 are on appeal in this case. Claims 1–28 are rejected under 35 U.S.C. 112, first paragraph, asserting that the specification fails to describe the claimed subject matter in such a way to enable one skilled in the art to make and/use the invention.

Claims 1–7 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hankinson et al. (USPN 6,799,202) ("Hankinson") in view of Toporek et al. (USPN 6,654,344) ("Toporek").

Claims 12–15, 17–21, 23 and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Abjanic (USPN 6,732,175) in view of Toporek.

Claims 25–28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Baskey (USPN 6,732,269) in view of Toporek.

STATUS OF AMENDMENTS

No amendments have been filed subsequent to the Final Office Action mailed April 20, 2006 from which this Appeal has been made.

SUMMARY OF THE CLAIMED SUBJECT MATTER

A concise summary of independent claims 1, 12 and 25 is provided below with reference to the specification and figures.

Independent claim 1

Claim 1 is directed to *a load balancing acceleration device comprising a processor, memory and communications interface*. FIGURE 3 and pp. 10-11 describe a secure socket layer (SSL) acceleration device. Pg. 10, ll. 25–28 describe the SSL acceleration device as having network interface hardware, random access memory and a microprocessor.

Claim 1 requires *a TCP communications manager capable of interacting with a plurality of client devices and server devices simultaneously via the communications interface*. Pg. 6, ln. 31 – pg. 7, ln. 2 states that the SSL acceleration device includes a TCP communications manager capable of interacting with a plurality of client devices and server devices simultaneously. Figure 3 shows the SSL acceleration device between a web client and a web server. Pp. 10–11 describes the SSL acceleration device of Figure 3 in communication with a web client and a web server.

Claim 1 requires *a secure communications manager to negotiate a secure communication session with one of the client devices*. Pg. 7, ll. 2–3, states that the SSL acceleration includes a secure communications manager. Originally filed claim 3 stated that secure communications manager of the SSL acceleration device negotiated a secure communications session with each of said plurality of client devices over an open network. Pg. 10, ll. 6–8 states that the SSL accelerator communicates with the web client using a secure protocol by communicating via a secure port 443 (shown in Figure 3 as the HTTPS protocol).

Claim 1 requires *an encryption and decryption engine instructing the processor to decrypt data received via the secure communication session and direct the decrypted data to one of said server devices via a second communication session*. Pg. 10, ll. 8–10 states that, as shown in Figure 3, the SSL accelerator will intercept data from the web client that is destined for port 443 of the web server, perform the SSL encryption and decryption, and forward the packets on to its destination, i.e., the web server.

Claim 1 requires *a load balancing engine associating each of said client devices with a respective one of said server devices based on calculated processing loads of each said server devices.* Originally filed claim 1 states that the SSL acceleration device includes a load balancing engine associating ones of said client devices with ones of said servers for a communications session based on calculated processing loads of each said server. Pg. 10, ll. 15–19 states that the SSL acceleration device of Figure 3 supports a number of operational modes of encryption and decryption, including a load balancing mode. Pg. 18, ln. 18 – Pg. 21 and Figure 6 describe in detail the SSL acceleration device when implementing load balancing amongst a number of servers.

Claim 1 requires that the decryption engine and the load balancing engine *bypass an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack.* Pg. 10, ll. 8–10 states that the SSL accelerator of Figure 3 performs the SSL encryption and decryption at the packet level rather than the transmitting packets up and down the TCP/IP stack as shown in Figure 2B before outputting the decrypted data to the server. Figure 6 and pg. 18, ln. 28 – pg. 21, ln. 19 describe the SSL accelerator as further supporting a load balancing mode in which the SSL acceleration device load balances the packets amongst a number of servers. Pg. 19, ll. 5–6 describes the SSL accelerator as load balancing between the servers by altering the source and destination addresses of packets. Figure 9A shows an overview of the various modes which may be implemented by the SSL device. As shown in Figure 9A and described on pg. 25, ll. 1–5, the SSL acceleration device can operate in a cut-through communication or a full proxy mode when performing SSL functions, and in each mode the load balancing (address redirection) scheme may be utilized.

Independent claim 12

Claim 12 recites *a method for performing acceleration of data communications between a plurality of customer devices attempting to communicate with an enterprise having a plurality of servers*. Figure 1 shows a Web client 100 coupled to the Internet 50 that may be coupled via a router 75 to an SSL accelerator device 250. The SSL accelerator device 250 is coupled to a plurality of Web servers 300.

Claim 12 requires *providing an intermediate acceleration device enabled for secure communication with the customer devices, wherein the acceleration device has an IP address associated with the enterprise*. Figure 3 and pp. 10–11 describe an SSL acceleration device shown intermediate a web client and a web server. Pg. 10, ll. 6–8 states that the SSL accelerator communicates with the web client using a secure protocol (shown in Figure 3 as the HTTPS protocol). Pg. 7, ln. 14 and original claim 2 describe the SSL acceleration device as having an IP address associated with the enterprise.

Claim 12 requires *receiving with the acceleration device communications directed to the enterprise in a secure protocol from one of the customer devices*, and *decrypting data packets of the secure protocol with the acceleration device to provide decrypted packet data*. Pg. 10, ll. 8–10 describes the SSL accelerator intercepting data from the web client using a secure protocol (HTTPS), and performing SSL encryption and decryption on the data.

Claim 12 requires, *selecting with the acceleration device at least one of the plurality of servers in the enterprise based on a load calculation including processing sessions of other servers in the enterprise and associating the selected server with a communications session from the one of the clients*. Originally filed claim 1 states that the SSL acceleration device includes a load balancing engine associating ones of said client devices with ones of said servers for a communications session based on calculated processing loads of each said server. Pg. 10, ll. 15–19 states that the SSL acceleration device of Figure 3 supports a number of operational modes of encryption and decryption, including a load balancing mode. Pg. 18, ll. 18 – Pg. 21 and Figure 6 describe in detail the SSL acceleration device when implementing load balancing amongst a number of servers.

Claim 12 requires that the SSL acceleration device perform the selection of the server from the plurality of servers (i.e., the load balancing) *without processing the data packets with an application layer of a network stack.* Pg. 10, ll. 8–10 states that the SSL accelerator of Figure 3 is capable of encrypting and decrypting communications at the packet level rather than the transmitting packets up and down the TCP/IP stack as shown in Figure 2B before outputting the decrypted data to the server. Figure 6 and pg. 18, ln. 28 – pg. 21, ln. 19 describe the SSL accelerator as load balancing the packets amongst a number of servers by altering the source and destination addresses of packets. Figure 9A shows an overview of the various modes which may be implemented by the SSL device. As shown in Figure 9A and described on pg. 25, ll. 1–5, the SSL acceleration device can operate in a cut-through communication or a full proxy mode when performing SSL functions, and in each mode the load balancing (address redirection) scheme may be utilized.

Claim 12 requires *forwarding the decrypted packet data from the acceleration device to the selected server of the enterprise.* Pg. 10, ll. 8–10 states that the SSL accelerator performs the SSL encryption and decryption and forwards the packet on to its destination, e.g., the web server. Figure 6 shows the load balancing mode of the SSL acceleration device, and block 270 of Figure 6 shows the SSL device as performing SSL functions, translating the destination address of the packets, and then forwarding the packets to the selected server.

Independent claim 25

Claim 25 is directed to *a system comprising a client device, a plurality of server devices, and an intermediate device coupled between the client devices and the server devices.* Figure 1 shows a Web client 100 coupled to the Internet 50 that may be coupled via a router 75 to an SSL accelerator device 250. The SSL accelerator device 250 is coupled to a plurality of Web servers 300. Figure 3 and pp. 10–11 describe an SSL acceleration device shown intermediate a web client and a web server.

Claim 25 requires that the intermediate device *intercept a request from the client device for a secure communication session and, in response to the request establish a secure communication session with the client device.* Pg. 10, ll. 8–10 describes the SSL accelerator intercepting data from the web client using a secure protocol (HTTPS). Figure 5 shows the SSL accelerator as intercepting a request for a session (block 204) and, in response, establishing a secure session with the client (block 210).

Claim 25 requires that the SSL acceleration device *selects one of the server devices based on resource loading experienced by the server devices without processing the request with an application layer of a network stack, and establishes a non-secure communication session with the selected server device.* Pg. 10, ll. 15–19 states that the SSL acceleration device of Figure 3 supports a number of operational modes of encryption and decryption, including a load balancing mode. Pg. 18, ln. 18 – Pg. 21 and Figure 6 describe in detail the SSL acceleration device when implementing load balancing amongst a number of servers. Pg. 10, ll. 8–10 states that the SSL accelerator of Figure 3 is capable of encrypting and decrypting communications at the packet level rather than transmitting packets up and down the TCP/IP stack as shown in Figure 2B before outputting the decrypted data to the server. Figure 6 and pg. 18, ln. 28 – pg. 21, ln. 19 describe the SSL accelerator as load balancing the packets amongst a number of servers by altering the source and destination addresses of packets. Figure 9A shows an overview of the various modes which may be implemented by the SSL device. As shown in Figure 9A and described on pg. 25, ll. 1–5, the SSL acceleration device can operate in a cut-through communication or a full proxy mode when performing SSL functions, and in each mode the load balancing (address redirection) scheme may be utilized.

GROUNDS OF REJECTION TO BE REVIEWED ON APPEAL

The first grounds for rejection to be reviewed on Appeal is the rejection of claims 1–28 under 35 U.S.C. 112, first paragraph, asserting that the specification fails to describe the claimed subject matter in such a way to enable one skilled in the art to make and/use the invention.

The second grounds for rejection to be reviewed on Appeal is the rejection of claims 1–7 and 22 under 35 U.S.C. 103(a) as being unpatentable over Hankinson et al. (USPN 6,799,202) ("Hankinson") in view of Toporek et al. (USPN 6,654,344) ("Toporek").

The third grounds for rejection to be reviewed on Appeal is the rejection of claims 12–15, 17–21, 23 and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Abjanic (USPN 6,732,175) in view of Toporek.

The fourth grounds for rejection to be reviewed on Appeal is the rejection of claims 25–28 under 35 U.S.C. 103(a) as being unpatentable over Baskey (USPN 6,732,269) in view of Toporek.

ARGUMENTS

The First Ground of Rejection

Claims 1–28 are on appeal in this case. All claims are rejected under 35 U.S.C. 112, first paragraph, asserting that the specification fails to describe the claimed subject matter in such a way to enable one skilled in the art to make and/use the invention. Appellants submit that, with respect to the first grounds of rejection, claims 1–28 stand or fall together.

With respect to claim 1, in the Final Office Action, pg. 2, the Examiner stated that he could not find support in the specification for an acceleration device in which a decryption engine and a load balancing engine "bypass an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack." With respect to claims 12 and 25 the Examiner indicated that he could not find support within the specification for the limitation "without processing the data packets with an application layer of a network stack."

Appellants submit that specification describes the claimed subject matter in such a way to enable one skilled in the art to make and/use the invention. Specifically, with respect to claim 1, the specification describes the claimed subject matter in such a way to enable one skilled in the art to make and use a load balancing acceleration device that bypasses an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack.

Prior to discussing Appellants' claimed acceleration device, Appellants refer the Board to Figure 2B and pages 5 and 6 of the present application that describe prior art SSL acceleration devices. For the convenience of the Board, Figure 2B is reproduced below:

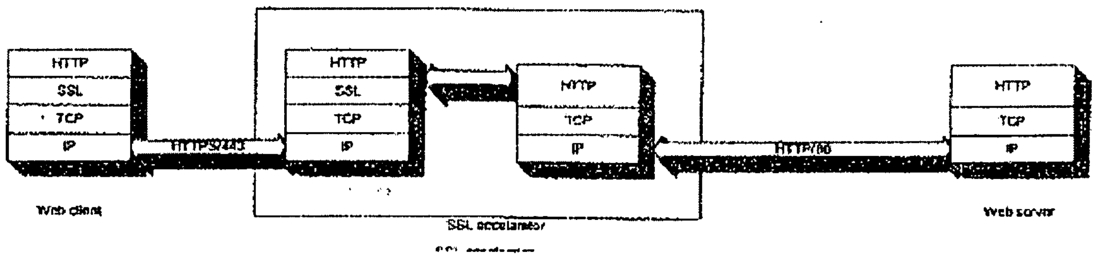

Figure 2B

The present application describes how the prior art SSL acceleration device of Figure 2B receives encrypted SSL packets from the web client and processes the data up the network stack through the session layer (SSL) all the way to the application layer (HTTP), i.e., layer seven of the OSI network stack. The prior art SSL accelerator then processes data back down the network stack to forward decrypted packets to the web server. That is, secure data received by the prior art SSL acceleration device of Figure 2B is processed up and down the networking stack through the following OSI network layers: IP→TCP→SSL→HTTP→HTTP→TCP→IP, where the top layer (i.e., Hypertext Transport Protocol) is well recognized as the application layer of the network stack.

Specifically the present application describes the prior art SSL accelerator of Figure 2B as follows:

> Figure 2B illustrates how SSL functions in the Open Systems Interconnect (OSI) Reference Model and in typical accelerators. The web client transmits data to the accelerator 250 in an encrypted form to the secure port 443 of the accelerator. In the client, the application layer protocol hands unencrypted data to the **session layer**; SSL encrypts the data and hands it down through the layers to the network IP layer, and on to the physical layers (now shown). Normally, a server will receive the encrypted data and when the server receives the data at the other end, it passes it up through the layers to the session layer where SSL decrypts it and hands it off to the application layer (HTTP). **The same happens in the typical SSL accelerator within the accelerator, where the data is handed to the <u>application layer</u>, processed, then returned down the stack from the HTTP layer to the IP layer for transmission to port 80 (in the clear) on the server coupled to the SSL accelerator.** Once at the server, the data returns up the stack for processing in the application layer. Since the client and the SSL device have gone through the key negotiation handshake, the symmetric key used by SSL is the same at both ends.

> In essence, the HTTP packet must travel through the TCP stack four times, creating a latency and CPU overhead and requiring full TCP stack support in the accelerator . . . .[1]

Thus, the present application identifies shortcomings of prior art SSL acceleration devices that require the device to process intercepted communications up the network stack to the application layer and then back down the stack. In the example of FIGURE 2B, the hypertext transfer protocol (HTTP) is an application-layer protocol, and SSL is implemented on blocks of application data above the packet level (e.g., in the session layer). Both of these points are made clear in the portion of the present application reproduced above.

Therefore, one skilled in the art would easily recognize that the prior art SSL accelerator (middle device) in Figure 2B process data through the <u>entire</u> networking protocol stack through the session layer (SSL) and up to and including the application layer (HTTP layer) and then back down the stack in order to provide network acceleration or load balancing. Application data is reassembled from the HTTP packets at the application layer. As explained in Appellants' Background, the HTTP packets travel through the entire TCP stack, creating a latency and CPU overhead and requiring full TCP stack support in the accelerator. This also requires a great deal of random access memory, usually around 8–10 kB per TCP session, for retransmission support. This type of architecture also has scalability and fault tolerance problems because all of the TCP and SSL state databases are concentrated on one SSL accelerator device.

After describing the prior art, the present application then describes in detail an acceleration device that decrypts the data from the secure communication sessions of the clients and outputs the decrypted data to the associated server devices without processing the data with the application layer of the network stack. For example, pg. 10, ll. 4–6 states that Figure 3 shows how the acceleration device of the present invention differs in general from that of the prior art of Figure 2B, and illustrates the manner in which the SSL encryption and decryption proxy is implemented. The present application continues, at pg. 10, ll. 8–10, by stating that, as shown in Figure 3, the SSL accelerator will intercept

[1] Present application, pg. 5, ln. 16–pg. 6, ln. 6 (emphasis added).

data from the web client that is destined for port 443 of the web server, perform the SSL encryption and decryption, and forward the packets on to its destination, i.e., the web server. Pg. 10, ll. 10–12 specifically states that "rather than the transmitting packets up and down the TCP/IP stack as shown in Figure 2B, [the SSL accelerator of Figure 3] will perform the SSL encryption and decryption at the packet level before forwarding the packet on to its destination." For the convenience of the Board, Figure 3 is reproduced below:

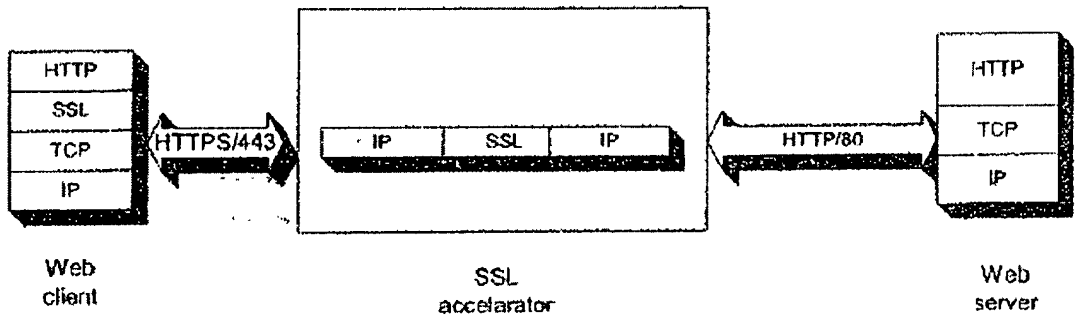

FIGURE 3

Appellants direct the Board's attention to the network stack shown in the SSL accelerator of Figure 3 where, in contrast to the prior art device of Figure 2B, the application layer (e.g., the HTTP protocol) of the network stack is noticeably absent when compared with the prior art accelerator shown in Figure 2B. Figure 3 clearly shows that, in that embodiment, Appellants' SSL accelerator does not process the secure data received from the web client at the application layer (e.g., HTTP) prior to forwarding the packets to the web server. Rather, as quoted above, the present application teaches that the accelerator shown in Figure 3 intercepts data destined for the web server and, rather than the transmitting packets up and down the TCP/IP stack as shown in Figure 2B, will perform the SSL encryption and decryption <u>at the packet level</u> before forwarding the packet on to its destination.[2] Therefore, one of ordinary skill would appreciate, based on the discussion and illustration of Figure 3, an example of Appellants' SSL accelerator performs SSL encryption and decryption by only processing the secure data received by

[2] Specification, pg. 10, ll. 8–12.

the SSL acceleration device through the following layers: IP→ SSL→ IP as shown in Figure 3. One of ordinary skill would recognize that Figure 3 clearly shows that the packets are passed up from the IP layer to the SSL layer (i.e., the session layer) and then back down to the IP layer for output. Based on this discussion, one of ordinary skill would easily recognize that the limitation of claim 1 of "bypass an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack" accurately states the described features of the described accelerator of Figure 3. For these reasons, Appellants are actually unsure why the Examiner stated that he was unable find support for this claim limitation, and the Examiner did not provide any analysis as to why he concluded that the present application failed to enable this claim limitation.[3]

As further evidence of its enabling teaching, the present application continues by describing in detail three distinct modes supported by the SSL acceleration device that utilize this "cut through" approach: (1) a direct, cut through processing method of Figure 5 in which packets from client to server are addressed from the client to the server and from server to client, with the intermediary, SSL device being transparent to both, (2) a cut though, load balancing approach of Figure 6 where the SSL device acts as a proxy and load balances amongst a number of servers, and (3) a full proxy mode of Figure 7, wherein the SSL device acts as a proxy for one or more servers, and handles both the SSL and TCP communications for the server.

These three modes are described in great detail on pp. 13–30 (eighteen pages) of the present application, including description of detailed flowcharts showing how these various modes implement SSL within the accelerator without requiring that application data be reassembled at the application layer. As just one example of the detailed technical disclosure within the present application, pg. 17, ll. 3–28 states:

> The SSL accelerator 250 includes a TCP/SSL session database to track all communication sessions occurring through it. Each session will have one or more records associated with it, with each record comprising an association of the TCP session sequence and the SSL sequence. Hence, on receiving the initial SYN

[3] See, Advisory Action, pg. 2, where the Examiner stated he could not find support for this element of claim 1.

from client 100 at step 202, the SSL accelerator will create a database entry for the particular session, associating the TCP-SSL sequence number pairs. The data may be considered as a table, with each row in the table representing one entry in a given session. Hence, for each session, a typical record might include up to about 8 – 16 records, which include a TCP sequence number, SSL session number, an initialization vector (for DES and 3DES) and an expected ACK.

During decryption, the device may utilize portions of its memory to buffer segments as necessary for decryption. The number and size of the buffers will depend on the cipher scheme used and the configuration of the packets, as well as whether the packets contain application data spanning multiple packets, referred to herein as multi-segment packets (and illustrated with respect to Figure 8). The SSL device can allocate SSL buffers as necessary for TCP segments. If, for example, application data having a length of 3000 bytes is transmitted via TCP segments having a length of 100 bytes, the device can, copy TCP segment 1 to a first SSL buffer, and start a timer, wait for packet 2 and when received, copy it to an SSL buffer and restart the timer, and finally when packet 3 is received, the SSL accelerator will copy it, decrypt all application data, authenticate it and forward the data on in the clear. (An alternative, bufferless approach is described below).

This section describes an example of how the SSL accelerator uses a TCP/SSL session database to track sessions, and TCP segments are buffered directly within SSL buffers and decrypted. Notably, an application-layer protocol, such as HTTP, is avoided.

As additional evidence, Figure 9A shows an overview of the various modes which may be implemented by the SSL accelerator. As shown in Figure 9A and described on pg. 25, ll. 1–5, the SSL acceleration device operates in cut-through communication (to avoid the upper levels of the network stack) either in a direct mode or a full proxy mode when performing SSL functions (i.e., the encryption and decryption operations). According to pg. 25, ll. 1–5, in each of these modes, the load balancing (address redirection) scheme may be utilized by the SSL acceleration device. Thus, the present application specifically states that the SSL acceleration device supports a load balancing mode in which the decrypted secure data is distributed amongst a plurality of servers. Figure 6 shows, and pg. 18, ln. 28 – pg. 21, ln. 19 describes, the SSL accelerator using the cut through approach in a load balancing mode in which the SSL acceleration device load balances the decrypted data packets amongst a number of servers. Pg. 19, ll. 5–6 provides enabling discussion that describes the SSL accelerator as load balancing between the servers by altering the source and destination addresses of packets, such as by using techniques commonly applied for Network Address Translation (NAT).

For at least these reasons, one of ordinary skill would appreciate that the present application enables an acceleration device having a decryption engine and a load balancing engine that "bypass an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack." Similarly, with respect to claims 12 and 25, the present application enables an SSL acceleration device that performs SSL encryption and operation, and selects a destination server "without processing the data packets with an application layer of a network stack."

Appellants' specification describes the claimed subject matter in such a way to enable one skilled in the art to make and use the invention. The Examiner has offered no evidence or analysis to support his statement that such elements are not supported by the present application. The rejection under 35 USC 112, first paragraph, should be withdrawn.

<u>The Second Ground of Rejection</u>

Claims 1–7 and 22 are rejected under 35 U.S.C. 103(a) as being unpatentable over Hankinson et al. (USPN 6,799,202) ("Hankinson") in view of Toporek et al. (USPN 6,654,344) ("Toporek"). Claim 1 requires an acceleration device in which a decryption engine and the load balancing engine bypass an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack.

Hankinson describes techniques for implementing a distributed, high capacity, high speed operating system.[4] According to Hankinson, the operating system may be incorporated into a web server having a plurality of "members." Each "member" has a distinct specialized operating system that is optimized for its function.[5] Thus, taken as a whole, Hankinson describes a web server in which multiple operating systems are used to

[4] Hankinson at Summary.
[5] *Id.*

perform functions, and a function may be implemented by operating systems executing on one or many different servers.

With respect to load balancing, Hankinson describes how the load of performing a networking function may be distributed across the different operating systems of the web server. Hankinson provides the example of a TCP/IP state machine, where execution of the state machine itself is distributed across the operating systems. That is, each of the plurality of operating systems of the web server cooperates to provide the TCP/IP function. Similarly, with respect to encryption, Hankinson notes that a member (i.e., an operating system) may support SSL. There is no teaching or suggestion that SSL is supported in a manner that is different from the prior art, i.e., where application data is reassembled via the application layer for decryption (see Figure 2B of Appellants' Background).

With respect to claim 1, the Examiner correctly recognizes that Hankinson provides no teaching or suggestion of a decryption engine and a load balancing engine that bypass an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack. Instead, the Examiner cites Toporek as teaching these elements and providing the motivation to modify the Hankinson web server because "it allows the network layer to communicate directly to the physical layer."[6]

Hankinson in view of Toporek fails to establish a prima facie case of obviousness for several reasons. First, even in combination, Hankinson in view of Toporek fail to provide any teaching relevant to providing encryption and decryption of secure data within an acceleration device in a manner that bypasses the application layer. Toporek describes a mechanism for controlling data flow from a satellite. The portion of Toporek cited by the Examiner describes a satellite gateway that merely *relays* information between a client and a server. The related information is not processed by the satellite gateway other than to control the rate of flow through the link. Toporek indeed states that the relayed information can flow through the satellite gateway at the network layer and

[6] Office Action, pg. 5, where the Examiner cites Toporek at col. 11, ll. 22–33.

bypass the transport and application layers, but this is merely because the data is not processed whatsoever. There is no teaching in Hankinson in view of Toporek as to how a device could decrypt secure data using a process that bypasses the application layer. Even if the Examiner's characterization of Toporek is correct that Toporek "allows the network layer to communicate directly to the physical layer," the combination of references still provides no teaching as to how the application-layer (which is above both the network layer and the physical layer) could be avoided for functions like SSL that traditionally require the application-layer.[7]

More specifically, Appellants' claim 1 requires decrypting the data from the secure communication sessions of the clients without processing the data with the application layer of the network stack. Hankinson describes a distributed operating system and makes only a passing reference to SSL, and the Toporek device only *relays* information. This combination provides no teaching whatsoever as to how a secure client communication could be decrypted, such as SSL. In fact, the combination does not suggest that SSL would be processed any differently whatsoever. According to Toporek, satellite communications that need not be processed can be relayed without processing. According to Hankinson, network components can be distributed to multiple devices.

The Examiner's conclusion of obviousness is based entirely on the unsupported assumption that the Hankinson in view of Toporek could somehow achieve an acceleration device capable of decrypting secure client communications without processing the secure communication at the application layer. It is important for the Examiner to appreciate that SSL and other secure communications of application data encrypt blocks of application data to form secure records above the packet level. Once formed, the secure records are passed down the network stack to the network layer where the secure records are split into packets. Decrypting SSL records without processing the records at an application layer is a non-trivial problem that is not remotely answered or suggested by any of the references. As one example, handling SSL communications of encrypted application data may require reassembly of secure records of application data that span multiple packets. Techniques for addressing these and other issues associated

[7] Office Action, pg. 5, where the Examiner cites Toporek at col. 11, ll. 22–33.

with secure client communications without processing the packets at the application layer are described throughout the application. See, e.g., pp. 17 and 26, which describe techniques by which Appellants' acceleration device is able to decrypt SSL records that span multiple packets without processing the packets at the application layer.

With respect to encryption, Hankinson merely notes that the system may support SSL. Hankinson provides no suggestion that SSL is supported in any manner that is different from the prior art. Toporek provides no mention of SSL or decrypting secure communications whatsoever. Therefore, there is no reasonable expectation that the Hankinson web server could be successfully modified in view of Toporek so as to somehow incorporate the function of decrypting data from a communication session without processing the data at the application layer. There is a significant gap in the teachings of the references as to how such a feature may even be implemented. The fact that Hankinson makes a passing reference to SSL and that Toporek describes a mechanism for relaying packet information without processing the packets whatsoever provides no enabling description of how encrypted communications could be decrypted with an acceleration device without processing the secure data at the application layer.

Appellants acknowledge that devices were known that simply relay packets without processing them at the application layer. Routers, for example, were such devices known at the time of Appellants' invention. However, no evidence has been introduced in the record to suggest that techniques were know for decrypting secure client communications, such as SSL, within an intermediate device without processing the secure communications at the application layer, and then forwarding application data to a server via a second communication session. Any modification to Hankinson in view of Toporek, as suggested by the Examiner, would at best achieve a web server capable of relaying information without processing the information at the application layer. Even when viewed in combination, Hankinson and Toporek provide no solution as to how to handle and decrypt an SSL or other secure communication session with an intermediate device without processing the secure data at the application layer.

Furthermore, the Hankinson federated operating system is not load balancing client devices with respect to server devices, as required by Appellants' claim 1. Rather, the federated operated system distributes the load of performing a **processing task**, such as implementing TCP/IP or implementing the function of SSL using the different operating systems and different computing resources. Toporek makes no mention of load balancing packets across servers, and merely describes relaying packets without processing the packets at the application layer. The Examiner's suggestion that the Hankinson operating system that load balances processing task could somehow be modified in view of Toporek to load balance decrypted data without processing the decrypted data at the application layer again glosses over significant gaps in the teachings of the references.

The Third Ground of Rejection

Claims 12–15, 17–21, 23 and 24 are rejected under 35 U.S.C. 103(a) as being unpatentable over Abjanic (USPN 6,732,175) in view of Toporek. Claim 12 requires, without processing the data packets with an application layer of a network stack, selecting with the acceleration device at least one of the plurality of servers in the enterprise based on a load calculation including processing sessions of other servers in the enterprise and associating the selected server with a communications session from the one of the clients.

Abjanic describes a network apparatus located between a network and a plurality of processing nodes or servers. The Abjanic network apparatus includes a content-based message director (e.g., *XML* director) to route or direct messages received from the network to one of the processing nodes *based upon the application data*, including business transaction information.[8]

In direct contrast to the elements of claim 21, Abjanic makes clear that the described apparatus is an application-layer content-based switching apparatus. Abjanic provides numerous examples of how application layer data (XML data in this case) is extracted using the HTTP protocol (which is an application-layer protocol) in order to make forwarding decisions. Below is one brief example:

> *The application data is provided after the HTTP header, and in this example is provided as XML data. . . . [T]he present invention is directed to a technique to perform switching at a network apparatus based upon the application data, such as XML data (which includes business transaction information).*[9]

[8] Abjanic at Summary.
[9] Abjanic at col. 6, ll. 1–25.

HTTP headers and XML data are only available at the application layer. Abjanic fails to teach or suggest mechanisms by which a load balancing acceleration device can, without processing the data packets with an application layer of a network stack, selecting with the acceleration device at least one of the plurality of servers in the enterprise based on a load calculation including processing sessions of other servers in the enterprise and associating the selected server with a communications session from the one of the clients.

The Abjanic appliance requires assembly of application data in order to make switching decisions. To address this deficiency, the Examiner summarily asserts that it would be obvious to modify the Abjanic appliance in view of the relay function of Toporek to perform such switching without processing decrypted data with an application layer.

However, the combination of Abjanic in view of Toporek fails to provide any teaching whatsoever as to, without processing the data packets with an application layer of a network stack, selecting with the acceleration device at least one of the plurality of servers in the enterprise based on a load calculation including processing sessions of other servers in the enterprise and associating the selected server with a communications session from the one of the clients.

In fact, similar to the Examiner's reasoning with respect to claim 1, the combination of Abjanic in view of Toporek does not suggest that secure communications would be processed any differently whatsoever. For example, Abjanic specifically requires application layer data in order to implement load balancing functions. According to Toporek, satellite communications that need not be processed can be simply relayed at lower levels of the stack. Even in combination, the Abjanic switch would still need to assemble XML data so as to make switching decisions, and the only teaching to this effect in Abjanic in view of Toporek requires assembly of application data at the application layer. The relay functions of Toporek do nothing to overcome or change this basic operation of Abjanic.

For these reason, there is no reasonable expectation of success for such a modification, as is required for a proper rejection under 35 USC 103. Such a modification so as to bypass the application layer, as proposed by the Examiner, would likely render the Abjanic device inoperable or defeat its essential purpose since no XML data would be identified. This is impermissible when forming a rejection under 35 USC 103. The Abjanic device specifically relies on XML data in order to perform switching applications. There is no suggestion whatsoever that the XML data in Abjanic would be available if the application layer were bypassed, as suggested by the Examiner. XML is application-layer data, and the Examiner has pointed to no teaching in Abjanic or Toporek that would allow the Abajanic device to perform load balancing functions without the availability of application data.

The Fourth Ground of Rejection

Claims 25–28 are rejected under 35 U.S.C. 103(a) as being unpatentable over Baskey (USPN 6,732,269) in view of Toporek. Claim 25 requires a client device, a plurality of server devices, and an intermediate device coupled between the client devices and the server devices. Claim 25 further requires that the intermediate device intercepts a request from the client device for a secure communication session, and in response to the request, the intermediate device establishes a secure communication session with the client device, selects one of the server devices based on resource loading experienced by the server devices without processing the request with an application layer of a network stack, and establishes a non-secure communication session with the selected server device.

In general, Baskey describes an SSL proxy server 40 that acts as a proxy server for a transaction server 50. In col. 6, ll. 17–35, Baskey makes mention that the SSL proxy server 40 may serve multiple transaction servers 50. However, at col. 6, ll. 17–35, directly counter to Appellants' claims, Baskey states that the routing function 42 may be provided SSL functions in the "application layer" of a protocol as an application program that receives information from the SSL of a first protocol stack and retransmits the information to a second SSL connection over a *second protocol stack*. Thus, the Baskey approach explicitly requires that secure data travel two full networking stacks, including the application layer, and Baskey fails to describe any other mechanism for implementing the SSL proxy.

To address this deficiency, the Examiner again summarily asserts that it would be obvious to modify the Baskey device in view of the relay function of Toporek to implement the proxy performance of Baskey without processing decrypted data with an application layer.

Again, Appellants submit that the satellite relay function of Toporek, where packets are not even processed, provides no enabling teaching whatsoever as to how the Baskey device could be modified so that SSL functions could still be implemented yet in a manner that avoids the application layer.

Further, there is no reasonable expectation of success for such a modification, as is required for a proper rejection under 35 USC 103. In fact, such a modification would likely render the Baskey device inoperable or defeat its essential purpose, which is impermissible when forming a rejection under 35 USC 103. Baskey states that the routing function 42 may be provided in the *application layer* of a protocol. There is no suggestion whatsoever that SSL functions of Baskey could even work if the application layer were bypassed, as suggested by the Examiner. Again, the Examiner's reasoning is based entirely on the unsupported assumption that the Toporek process could be applied in some other context, namely loadbalancing and operation as a proxy server in the case of the Baskey reference. However, nothing in Baskey or Toporek bridges this significant gap as to how the Toporek packet relay function that bypasses both the transport and the application layer could possibly be used within a load-balancing, full proxy device, such as the Baskey device.

For at least these reasons, the references fail to establish a prima facie case for non-patentability of Appellants' claims under 35 U.S.C. 103(a). Withdrawal of this rejection is requested.

Conclusion of Arguments

The Examiner erred in rejecting Appellants' claims under 35 U.S.C. 112, first paragraph and 35 U.S.C. 103(a). Reversal of these rejections and allowance of the pending claims are requested.

Respectfully submitted,

Date:
December 20, 2006

By: [illegible]
Name: Jennifer M.K. Rogers
Reg. No.: 58,695

Shumaker & Sieffert, P.A.
8425 Seasons Parkway, Suite 105
St. Paul, Minnesota 55125
Telephone: (651) 735-1100 ext. 11
Facsimile: (651) 735-1102

APPENDIX: CLAIMS ON APPEAL

Claim 1 (Previously Presented): A load balancing acceleration device, comprising:

a processor, memory and communications interface;

a TCP communications manager capable of interacting with a plurality of client devices and server devices simultaneously via the communications interface;

a secure communications manager to negotiate a secure communication session with one of the client devices;

an encryption and decryption engine instructing the processor to decrypt data received via the secure communication session and direct the decrypted data to one of said server devices via a second communication session; and

a load balancing engine associating each of said client devices with a respective one of said server devices based on calculated processing loads of each said server devices,

wherein the decryption engine and the load balancing engine bypass an application layer of a network stack by decrypting the data from the secure communication sessions of the clients and outputting the decrypted data to the associated server devices without processing the data with the application layer of the network stack.

Claim 2 (Previously Presented): The device of claim 1 wherein the TCP communications manager provides an IP address of an enterprise to said secure communications manager, and each of said plurality of servers devices is associated with the enterprise.

Claim 3 (Previously Presented): The device of claim 2 wherein the secure communications manager negotiates a secure communication session with each of said plurality of client devices over an open network.

Claim 4 (Previously Presented): The device of claim 3 wherein the TCP communications manager negotiates a separate, open communications session with one of the plurality of server~~s~~ devices associated with the enterprise for each secure communication~~s~~ session negotiated with the ~~a~~ client devices based on the associations of said client devices to said server devices by said load balancing engines.

Claim 5 (Previously Presented): The device of claim 1 wherein the encryption and decryption engine decrypts the data on a packet level by decrypting packet data received on the communications interface via the secure communication~~s~~ session to extract a secure record, decrypting application data from the secure record in the packet data, and outputting the decrypted application data from the secure record to the one of said server devices via the second communication session without processing the application data with ~~an~~ the application layer of the network stack.

Claim 6 (Previously Presented): The device of claim 5 wherein the load-balancing engine selects the second communication session.

Claim 7 (Previously Presented): The device of claim 1 wherein the TCP communications manager responds to TCP communications negotiations directly for an enterprise.

Claim 8 (Previously Presented): The device of claim 1,

wherein the TCP communications manager receives packets from the client devices, and

wherein the TCP communications manager changes destination IP addresses for the packets to IP addresses for the server devices.

Claim 9 (Previously Presented): The device of claim 8,

wherein the TCP communications manager maintains TCP communication sessions with the server devices, and

wherein the secure communications manager negotiates a secure communication session for each TCP communications session.

Claim 10 (Original): The device of claim 9 wherein the secure communications manager responds to all secure communications with each client device.

Claim 11 (Previously Presented): The device of claim 9 wherein the secure communications manager changes a destination IP address for each packet to a server IP address.

Claim 12 (Previously Presented): A method for performing acceleration of data communications between a plurality of customer devices attempting to communicate with an enterprise having a plurality of servers, comprising:

providing an intermediate acceleration device enabled for secure communication with the customer devices, wherein the acceleration device has an IP address associated with the enterprise;

receiving with the acceleration device communications directed to the enterprise in a secure protocol from one of the customer devices;

decrypting data packets of the secure protocol with the acceleration device to provide decrypted packet data;

without processing the data packets with an application layer of a network stack, selecting with the acceleration device at least one of the plurality of servers in the enterprise based on a load calculation including processing sessions of other servers in the enterprise and associating the selected server with a communications session from the one of the clients; and

forwarding the decrypted packet data from the acceleration device to the selected server of the enterprise.

Claim 13 (Original): The method of claim 12 further including the steps of:

receiving application data from the selected server of the enterprise;

encrypting the application data received from the selected server; and

forwarding encrypted application data to the customer device.

Claim 14 (Previously Presented): The method of claim 12 wherein the step of receiving communications directed to the enterprise includes receiving with the device communications having a destination IP address of the enterprise.

Claim 15 (Previously Presented): The method of claim 12 further including the step of negotiating the secure protocol session with the customer device by responding as the enterprise to the customer devices.

Claim 16 (Previously Presented): The method of claim 12 further wherein the step of forwarding comprises:

modifying a destination IP address of data packets from an IP address associated with the enterprise IP to an IP address for the selected server.

Claim 17 (Previously Presented): The method of claim 12 wherein the step of forwarding comprises:

establishing an open communication session from the acceleration device to the selected server, and

mapping the decrypted packet data to the open communication session established with the selected server.

Claim 18 (Previously Presented): The method of claim 17 wherein the open communication session is established via a secure network.

Claim 19 (Previously Presented): The method of claim 12 wherein the step of receiving comprises:

receiving encrypted data having a length greater than a TCP segment carrying said data; and

wherein said step of decrypting comprises:

buffering the encrypted data in a memory buffer in the acceleration device, the buffer having a length equivalent to the block cipher size necessary to perform the cipher; and

decrypting the buffered segment of the received encrypted data to provide decrypted application data.

Claim 20 (Previously Presented): The method of claim 19 further including the step of authenticating the data on receipt of a final TCP segment on a packet level without processing the application data with the ~~an~~ application layer of the network ~~a TCP/IP~~ stack.

Claim 21 (Original): The method of claim 19 further including the step of generating an alert if said step of authenticating results in a failure.

Claim 22 (Previously Presented): The device of claim 1, wherein the device comprises a network router.

Claim 23 (Previously Presented): The method of claim 12, wherein decrypting data packets comprises decrypting the data packets at a packet level of the network ~~a TCP/IP~~ stack.

Claim 24 (Previously Presented): The method of claim 12, wherein decrypting data packets comprises:

decrypting the data packets to extract a secure record,

decrypting application data from the secure record, and

authenticating the application data without processing the application data with ~~an~~ the application layer of the network stack.

Claim 25 (Previously Presented): A system comprising:

a client device;

a plurality of server devices; and

an intermediate device coupled between the client devices and the server devices,

wherein the intermediate device intercepts a request from the client device for a secure communication session, and

wherein, in response to the request, the intermediate device establishes a secure communication session with the client device, selects one of the server devices based on resource loading experienced by the server devices without processing the request with an application layer of a network stack, and establishes a non-secure communication session with the selected server device.

Claim 26 (Previously Presented): The system of claim 25, wherein the intermediate device receives encrypted data from the client device via the secure communication session, decrypts the data and forwards the decrypted data to the selected server device via the non-secure communication session.

Claim 27 (Previously Presented): The system of claim 25, wherein the intermediate device receives unencrypted data from the selected server device via the non-secure communication session, encrypts the data and forwards the encrypted data to the client device via the secure communication session.

Claim 28 (Previously Presented): The system of claim 25, wherein the intermediate device comprises a network router.

APPENDIX: EVIDENCE

None

APPENDIX: RELATED PROCEEDINGS

None